LO ZX 80 CON LA NUOVA ROM

Dr.ssa **Rita Bonelli**

PREFAZIONE

È arrivata la nuova ROM 8K per il SINCLAIR ZX80 con la nuova maschera per la tastiera. Il nostro piccolo calcolatore si arricchisce, diventa un altro calcolatore, più potente e più versatile.

Lo scopo di questo manuale è quello di fornire tutte le informazioni necessarie per imparare ad usare lo ZX80 con la nuova ROM. Nelle pagine seguenti si farà sempre riferimento al manuale "IMPARIAMO A PROGRAMMARE IN BASIC CON LO ZX80".

La nuova ROM consente il trattamento di:

- numeri interi e decimali;
- variabili numeriche con più indici;
- variabili stringa con uno o più indici;
- parti di stringhe;
- funzioni matematiche e trigonometriche;
- funzioni di stringa;
- operatori di relazione compositi;
- funzioni grafiche sul video;
- files di programmi con nome su cassetta;
- una stampante.

Il nuovo BASIC ha anche delle funzioni non standard che suppliscono egregiamente alla mancanza di qualche funzione standard.

Nelle pagine seguenti vengono segnalate le novità rispetto alla vecchia ROM, non sono stati spiegati di nuovo tutti i comandi BASIC, ma nella Appendice C troverete la scheda completa del nuovo BASIC.

MONTAGGIO NUOVA ROM E MASCHERINA TASTIERA

L'operazione di sostituzion della ROM è molto semplice. Per facilitarla ulteriormente si consiglia di acquistare un "Estrattore" e un "Inseritore" della "OK TOOL", reperibili presso tutte le Sedi G.B.C. rispettivamente con i numeri di codice: SM/5265-00 e SM/5280-00.

Schematizziamo la procedura:

- 1) Estrarre le 5 clips che tengono chiuso il contenitore di plastica del calcolatore.
- 2) Togliere il coperchio di plastica mettendo allo scoperto i diversi componenti del calcolatore.
- 3) Togliere la vecchia ROM, facilmente riconoscibile dalla

Fig. 1 - Ecco come si presenta lo ZX80 dopo aver rimosso le 5 clips bianche che fissano il coperchio.

scritta ROM e situata nell'angolo destro in alto:

- o con l'attrezzo estrattore mediante una semplice pressione verso l'alto;
- o manualmente facendo leva con un piccolo cacciavite tra la ROM e lo zoccolo sottostante.

• 4) Inserire la nuova ROM:

- o con l'attrezzo inseritore, dopo avervi delicatamente inserito la nuova ROM, appoggiandolo sullo zoccolo rispettando la posizione della tacca ed esercitando una leggera pressione;
- o manualmente prendendo la ROM tra il pollice e l'indice ed inserendola nello zoccolo rispettando la posizione della tacca.
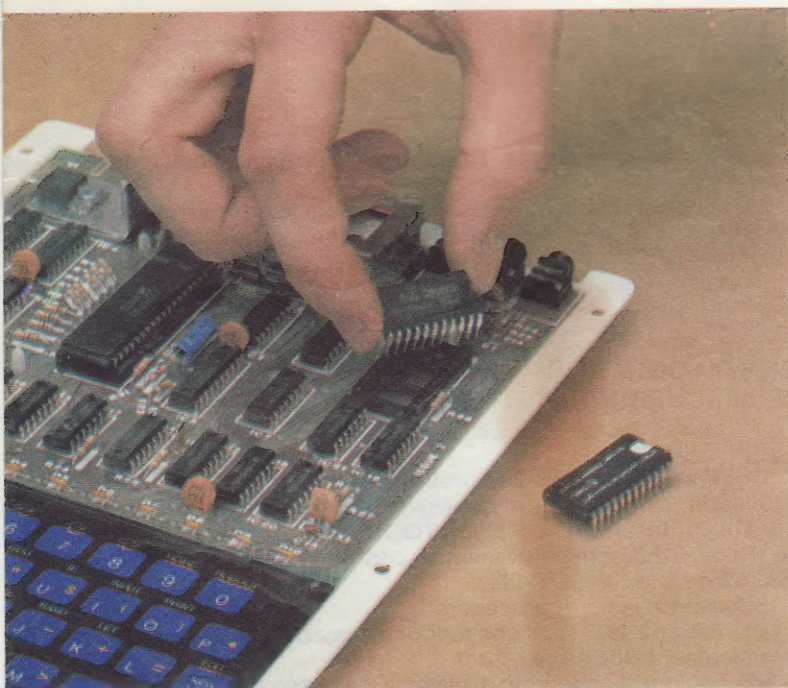
Fig. 2 - La delicata fase di sostituzione della ROM.

• 5) Togliere le 2 clips nere poste sul bordo inferiore della tastiera, applicare la nuova mascherina sopra la vecchia, facendo combaciare i fori per le clips, quindi rimettere le 2 clips nere.

• 6) Rimontare il coperchio di plastica e fissarlo con le 5 clips tolte inizialmente.

IL CURSORE DELLO SCHERMO

Restano valide tutte le informazioni fornite nel capitolo 8 del Manuale ZX80.

Il cursore dello schermo può contenere anche le lettere F e G, sempre in campo inverso.

La lettera F appare se si premono contemporaneamente i tasti SHIFT e FUNCTION, resta attiva solo per la pressione del tasto successivo e serve per selezionare le funzioni scritte sotto i tasti.

La lettera G appare se si premono contemporaneamente i tasti SHIFT e GRAPHICS, resta attiva fino a quando si premono di nuovo contemporaneamente questi due tasti e consente di selezionare:

- un carattere in campo inverso premendo il relativo tasto;
- un carattere grafico premendo il relativo tasto contemporaneamente al tasto SHIFT.

LA NUOVA TASTIERA

Nella nuova tastiera solo il tasto SHIFT reca una sola dicitura, esso serve:

- per attivare le funzioni scritte in rosso su tutti gli altri tasti;
- per ottenere i caratteri grafici;

e l'effetto prodotto dipende dallo stato del calcolatore.

Gli altri tasti hanno tutti più funzioni e queste vengono rese attive in dipendenza dallo stato del calcolatore, stato evidenziato nella lettera che appare in campo inverso sul cursore dello schermo, senza o con l'uso contemporaneo del tasto SHIFT.

Esaminiamo ciò che è scritto all'interno dei tasti; abbiamo:

- cifre, lettere, simboli o caratteri grafici in nero nella parte bassa;
- simboli o parole in rosso nella parte alta.

Le cifre, le lettere e i simboli vengono accettati quando il cursore dello schermo si trova nello stato L.

I caratteri grafici sono accettati quando il cursore si trova nello stato G (si passa a questo stato premendo contemporaneamente SHIFT e GRAPHICS) e si premono contemporaneamente il tasto SHIFT ed il tasto grafico.

Se il cursore si trova nello stato G e si preme un qualunque tasto, senza lo SHIFT si ottiene il carattere in campo inverso.

Per uscire dallo stato G e tornare allo stato L si devono premere ancora contemporaneamente SHIFT e GRAPHICS.

I simboli e le parole in rosso vengono accettati se si preme il tasto contemporaneamente al tasto SHIFT.

Le parole scritte sotto i tasti sono considerate funzioni e sono attive quando il cursore si trova nello stato F. Lo stato F si ottiene premendo contemporaneamente i tasti SHIFT e FUNCTION.

Le parole scritte sopra i tasti sono parole chiave del linguaggio BASIC e sono attive quando il cursore si trova nello stato K.

I comandi FAST e SLOW non sono usati.

LE VARIABILI E LE COSTANTI

La nuova ROM consente di usare numeri interi o decimali aventi almeno 9 cifre di precisione. Si arriva alle 10 cifre se i numeri si mantengono inferiori a 4294967296.

Il calcolatore accetta dalla tastiera numeri scritti in 3 modi:

- 1) numeri interi;
- 2) numeri con il punto decimale;
- 3) numeri in notazione esponenziale.

Per quanto riguarda i punti 1) e 2) non è necessario fornire spiegazioni, basta solo ricordare che, usando i calcolatori elettronici, il punto decimale sostituisce la virgola decimale.

Il punto 3) si riferisce ai numeri scritti sotto forma di prodotto di un numero per una opportuna potenza di 10.

Esempio:

5.27 = 527*10**-2 = 0.527*10**1 = 52.7*10**-1

È chiaro che l'esempio potrebbe essere modificato all'infinito e questo non avrebbe molto senso (ricordiamo che "* significa moltiplicato" e che "** significa elevato alla potenza di"). È invece interessante notare che ogni numero può essere scritto in forma esponenziale in modo univoco se si pongono tutte le cifre significative a destra del punto decimale, cioè "0.cifre" e si usa un opportuno esponente per il moltiplicatore 10. Questo modo di scrivere i numeri viene chiamato "forma esponenziale normalizzata".

Nella forma esponenziale normalizzata vengono conservate tutte le cifre a partire dalla prima cifra significativa (diversa da zero) e questo consente, usando un numero prefissato di cifre, di conservare sia i numeri molto grandi che i numeri molto piccoli con una precisione predeterminata. L'esponente serve poi a dare la grandezza reale del numero.

Inoltre non è necessario conservare "0.", ma basta conservare le cifre dopo il punto; esse prendono il nome di "mantissa". Analogamente non è necessario conservare "*10**", ma basta conservare l'esponente; esso prende il nome di caratteristica.

Per fare accettare dal calcolatore un numero in notazione esponenziale esso deve essere scritto così: numero Exxx

dove: numero — è il numero scritto come intero o come decimale e non necessariamente in forma normalizzata

E — corrisponde convenzionalmente a "*10**";

xxx — sta per un numero al massimo di 2 cifre con o senza segno e rappresenta l'esponente di 10.

Esempi:

0.527E1 che corrisponde a 5.27
527E-2 che corrisponde a 5.27
4.1E10 che corrisponde a 41000000000

Qualunque numero, non importa in quale modo sia stato immesso nel calcolatore, viene memorizzato in forma esponenziale normalizzata. Il sistema usa 5 bytes per memorizzare un numero:

- 1 byte serve per la caratteristica;
- 4 byte servono per la mantissa.

Ovviamente, dato che il sistema conserva i numeri in forma binaria, la caratteristica rappresenta l'esponente da dare al moltiplicatore 2 (e non 10) per ottenere il numero, rappresentato a sua volta da una mantissa binaria.

Il byte della caratteristica (il cui valore può andare da 0 a 255) viene usato come esponente dopo avergli sottratto 128; in tale modo gli esponenti positivi variano apparentemente da 129 a 255 e realmente da 1 a 127 e quelli negativi variano apparentemente da 0 a 127 e realmente da −128 a −1. L'esponente reale 0 corrisponde all'esponente apparente 128.

Fig. 3 - Applicazione della nuova mascherina dopo aver rimosso le clips nere.

Fig. 4 - Lo ZX80 dotato di nuova ROM, tastiera, espansione a 16K e piccolo altoparlante.

Fig. 5 - La nuova tastiera dello ZX80 dove si possono vedere le nuove funzioni e la particolarità di ogni tasto che arriva ad avere fino a cinque differenti funzioni.

Questo significa che la caratteristica dei numeri trattati varia in decimale da −39 a +38.

I 4 byte della mantissa servono per conservare una mantissa normalizzata usando le seguenti convenzioni:

- il primo bit del byte più alto serve per il segno: 0 per numeri positivi e 1 per numeri negativi;
- i 31 bits rimanenti servono per la mantissa, ma dato che, essendo essa normalizzata comincia sicuramente con un bit 1, questo primo bit viene omesso, si ha così un bit in più di precisione. Naturalmente nei calcoli viene tenuto conto anche di questo primo bit. Il più grande numero memorizzabile in 32 bits è = a 2**32-1, che è appunto il numero 4294967295. Questo è di 10 cifre, ma non è il massimo numero di 10 cifre disponibile (sarebbero 10 cifre 9 consecutive) e quindi si dice che la precisione è tra le 9 e le 10 cifre.

  I numeri negativi non sono nella forma di complemento a 2, ma il primo bit è uguale a 1 per indicare il segno meno, ed è seguito dal valore assoluto del numero privato del primo bit.

Il numero zero è rappresentato dai 5 bytes tutti al valore 0.

Quando i numeri vengono stampati si vedono solo 8 cifre significative eventualmente seguite da zeri e con l'ultima cifra significativa arrotondata, però il numero viene conservato in memoria con la precisione su esposta. Si possono provare queste caratteristiche del calcolatore scrivendo semplici esempi con calcoli di numeri molto grandi e molto piccoli.

Il nuovo BASIC del Sinclair consente quindi di trattare numeri interi o decimali senza le usuali distinzioni tra interi e decimali presenti nei Basic Standard, dove l'aggiunta di un suffisso al nome della variabile o alla costante crea una distinzione tra interi e decimali.

Per quanto riguarda la formazione dei nomi delle variabili sono ancora valide tutte le regole esposte nel Manuale dello ZX80.

Le variabili numeriche con indice consentono l'uso di dimensioni miltiple, cioè non si ha più un solo indice, ma quanti indici si vuole. Il nome delle variabili numeriche con indice deve essere formato da una sola lettera. Gli indici partono dal valore 1. Per dimensionare le variabili con indice si usa la frase DIM.

Si può usare lo stesso nome per una variabile numerica semplice e per una variabile con indice ed il sistema le considera diverse.

Esempio:

```
10   REM MATRICE DI 3 RIGHE E 4 COLONNE
15   DIM M (3,4)
20   FOR I = 1 TO 3
30   FOR K = 1 TO 4
40   LET M (I,K) = I*10+K
45   PRINT M (I,K); "   ";
50   NEXT K
55   PRINT
60   NEXT I
```

Le regole per la formazione dei nomi delle stringhe sono invariate, ma sono disponibili le variabili stringa con indice a dimensioni multiple, con la limitazione che tutti gli elementi stringa devono avere la stessa lunghezza. Nel dimensionamento di una matrice (variabile con indice di dimensioni multiple) di stringhe dopo le dimensioni deve essere presente un numero che definisce il numero dei caratteri di ogni elemento.

Esempio:

10 DIM A$(10,7)	definisce una matrice di stringhe ad una dimensione (vettore), formata da 10 elementi stringa di 7 caratteri ciascuno.
100 DIM B$(7,5,10)	definisce una matrice di stringhe a due dimensioni, formata da 7 righe e 5 colonne, quindi 35 stringhe, ognuna di 10 caratteri.

Non si può usare lo stesso nome per una variabile stringa semplice e per una variabile stringa con indice.

Gli indici possono essere: costanti, variabili, espressioni numeriche; essi vengono arrotondati all'intero più vicino.

Ricordiamo che le variabili semplici devono essere inizializzate per cominciare ad esistere in un programma cioè ci deve essere o una frase LET di assegnazione o la variabile deve comparire dopo la parola INPUT e quindi ricevere un dato al momento dell'esecuzione del programma.

Le variabili con indice invece cominciano ad esistere al momento della DIM e vengono inizializzate o a zero o con spazi.

TRATTAMENTO DELLE STRINGHE

Le variabili stringa ricevono un contenuto o con una frase LET di assegnazione o con una INPUT. Nel primo caso il dato deve essere contenuto tra doppi apici, nel secondo i doppi apici compaiono automaticamente sullo schermo.

Ovviamente l'unico carattere che non può far parte della stringa è il doppio apice (", chiamato QUOTE), ma si può ottenerlo, se desiderato, usando il carattere chiamato "QUOTE IMAGE" corrispondente ai tasti SHIFT e Q, il quale in fase di stampa appare come un doppio apice.

Si può usare l'operatore + per concatenare tra loro più stringhe, cioè

```
10   LET A$ = "GIORNATA"
20   LET B$ = "DI FESTA"
30   LET C$ = A$ + B$
40   PRINT C$
```

appare GIORNATA DI FESTA perchè C$ contiene le due stringhe A$ e B$ concatenate.

Nel seguito vengono elencate le funzioni che hanno attinenza con il trattamento delle stringhe. Per ognuna viene indicato il tipo dell'argomento; esso deve essere scritto tra parentesi se è una espressione. Se è una costante o una variabile può essere scritto senza parentesi.

Le funzioni disponibili per le stringhe sono:

- CHR$ (argomento numero)
  Fornisce il carattere corrispondente al codice numerico su cui opera. il codice deve essere compeso tra 0 e 255, altrimenti si ha errore.

Esempio:

```
10 FOR K=1 TO 26
20 PRINT CHR$ (K+37);
30 NEXT K
```

stampa le 26 lettere dell'alfabeto.

•CODE (argomento stringa)
Fornisce il codice numerico del primo carattere della stringa. Se la stringa è la stringa nulla ottenuta scrivendo due volte il doppio apice (da non confondere con il carattere SHIFT e Q) si ottiene 0. Esempio:

```
100 PRINT CODE ("OGGI")
```

stampa 52, codice della lettera O.

•LEN (argomento stringa)
Fornisce la lunghezza della stringa. Se applicata alla stringa nulla dà 0. Esempio:

```
20 LET X = LEN (A$)
```

se A$ ="PIOVE", pone in X il valore 5.

•STR$ (argomento numero)
Trasforma un numero o una espressione nella stringa corrispondente. Esempio:

```
10 LET C=-345
20 PRINT STR$ (C), STR$ (34+8.9)
```

stampa
—345

•VAL (argomento stringa)
Fornisce un numero corrispondente alla stringa che deve essere numerica, altrimenti si ha errore. Esempio:

```
20 LET A$ ="-345.8"
30 LET X=VAL (A$)
40 LET Z = X + 18
```

consente di operare un calcolo sul contenuto di A$

Non è più disponibile la funzione di stringa TL$, ma essa non è più necessaria potendo trattare una qualunque stringa come una variabile stringa con indice e quindi potendo accedere ad ogni carattere mediante un indice.

Si definisce SUBSTRINGA una qualunque porzione di STRINGA formata da caratteri consecutivi. Se consideriamo la stringa A$ ="FELICEMENTE", la stringa B$ ="MENTE" è una substringa di A$, mentre la stringa C$ ="LIMENTE" non lo è perchè non è formata da caratteri consecutivi di A$.

Nel nuovo BASIC c'è la possibilità di riferirsi a substringhe di una qualunque stringa.

Per ottenere la stringa B$ di cui sopra possiamo scrivere

```
100 LET B$ = A$(7 TO 11)
```

cioè prendiamo i caratteri di A$ dal settimo all'undicesimo.

Questo tipo di operazione prende il nome di "slicing". Si deve far seguire alla stringa dalla quale si vuole estrarre una parte una parentesi e porre entro la parentesi il numero d'ordine del carattere da cui iniziare l'estrazione, la parola chiave TO ed il numero d'ordine del carattere con il quale terminare l'estrazione. Uno dei due numeri o tutti e due possono mancare, come risulta dagli esempi seguenti, che non sono scritti nella forma di frasi BASIC, ma servono solo per spiegare la logica dell'operazione:

"PIPPO" (TO 5) = "PIPPO" (1 TO 5) = "PIPPO"
"PIPPO" (2 TO) = "PIPPO" (2 TO 5) = "IPPO"
"PIPPO" (TO) = "PIPPO" (1 TO 5) = "PIPPO"
"PIOVE" (2 TO 2) = "PIOVE" (2) = "I"
"PIOVE"(3 TO 7) dà errore, la stringa è di 5 caratteri
"PIPPO" (5 TO 4) = " ", cioè la stringa nulla.

I due numeri devono essere positivi, altrimenti si ha errore.

Il programma che segue toglie dalla stringa A$ tutti gli spazi di riempimento a destra, ottenendo una stringa B$, e poi stampa le due stringhe tra doppi apici.

```
10    INPUT A$
20    FOR N=LEN A$ TO 1 STEP -1
30    IF A$ (N) < >  "  " THEN GOTO 50
40    NEXT N
50    LET B$ = A$(TO N)
60    PRINT """";A$;"","""";B$;""""
70    GOTO 10
```

Alla linea 30 l'operazione di "slicing" consente di trattare i caratteri della stringa A$ come se essa fosse una stringa dimensionata con una DIM come variabile con indice. Alla linea 60 si fa uso del carattere "quote image" per ottenere la stampa delle due stringhe A$ e B$ tra doppi apici. Se la stringa A$ fosse tutta di spazi, alla linea 50 si arriverebbe con N=0 e quindi B$ risulterebbe la stringa nulla.

Se si opera su variabili stringa, ed ovviamente non su costanti, si possono anche modificare alcuni caratteri nella stringa, cioè operare una sostituzione invece di una estrazione. Esempio:

```
10    LET A$ ="SEI FELICE"
20    LET A$ (5 TO 10)="******"
30    PRINT A$
```

si ottiene: SEI ******

Se alla linea 20 la substringa sostitutiva è più lunga della parte da sostituire essa viene troncata.

L'operazione di "slicing" ha priorità 12.

L'operazione di "slicing" non è standard; essa è molto versatile e consente di supplire alla mancanza, in questo BASIC di funzioni di stringa come: LEFT$, RIGHT$ e MID$.

PRIORITA' OPERATIVA

Alle funzioni e alle operazioni è stata assegnata una priorità operativa codificandola mediante un numero. Il numero della priorità può variare da 16 a 1. Il numero più alto corrisponde alla più alta priorità, cioè all'operazione o funzione più importante rispetto alle altre. Nella valutazione delle espressioni vengono eseguite le operazioni da sinistra a destra eseguendo prima quelle che hanno più alta priorità, naturalmente dando la precedenza alle operazioni contenute nelle parentesi.

TABELLINA DELLE PRIORITA'

Operazione	Priorità
Indici e slicing	12
Tutte le funzioni eccetto NOT e - unitario	11
**(elevamento a potenza)	10
- unitario (segno numeri negativi)	9
*,/(moltiplicazione e divisione)	8
+,-(somma e sottrazione)	6
=,>,<,<=,>=,<>	5
NOT	4
AND	3
OR	2

OPERATORI RELAZIONALI E LOGICI

Gli operatori relazionali sono:

=	uguale
>	maggiore
<	minore
<=	minore o uguale
>=	maggiore o uguale
<>	diverso

la condizione logica di verità corrisponde al valore 1 della variabile logica, mentre la condizione di non verità corrisponde al valore 0.

Gli operatori logici sono rimasti invariati: NOT, AND e OR, ma anche qui la condizione vera corrisponde a 1 e la falsa a 0.

Le operazioni di relazione possono far parte di espressioni aritmetiche e nel calcolo ad esse viene sostituito il valore della corrispondente variabile logica.

LE FUNZIONI

Usando il tasto SHIFT insieme al tasto FUNCTION il cursore dello schermo passa allo stato F. Quando il cursore dello

schermo è nello stato F la pressione di uno dei 25 tasti che recano una funzione scritta sotto fa accedere alla corrispondente funzione e il cursore ritorna nello stato L.

In questo paragrafo si descrivono tutte le funzioni salvo le:

CHR$, CODE, LEN, STR$, VAL

di cui si è parlato nel paragrafo "Trattamento delle stringhe".

Tutte le funzioni meno tre richiedono un argomento che non è necessario porre tra parentesi se è una costante o una variabile, se invece l'argomento è una espressione esso va posto tra perentesi. Le due funzioni che non richiedono un argomento sono:

PI, RND e INKEY$

Le funzioni di tipo matematico danno una precisione di circa 10 cifre e mantengono tali cifre in memoria anche se ne mostrano solo 8 sul video.

ELENCO FUNZIONI

Funz.	argomento	Commento
ABS	numero	Valore assoluto.
ASN	numero	Arcoseno in radianti. Errore A se argom. non tra -1 e + 1.
AT	numeri	L'argomento è dato da due numeri separati da virgola: ATx,y, dove x e y rappresentano le coordinate del punto del video dove si vuole evidenziare il prossimo carattere. Il primo numero, x, si riferisce alla linea e può variare da 0 a 21. Il secondo numero, y, si riferisce alla colonna e può variare da 0 a 31. Questa funzione può essere usata nel comando PRINT. Si considera linea 0 la più alta e colonna 0 la più a sinistra.
ATN	numero	Arcotangente in radianti.
COS	numero in radianti	Coseno.
EXP	numero	Calcola "e" elevato al numero. e=2.718281828.
INKEY$	nessuno	Legge un carattere dalla tastiera, esso corrisponde al tasto premuto quando il cursore è nel modo L. Se non si preme alcun tasto si ha la stringa nulla.
INT	numero	Parte intera del numero troncato senza arrotondamento.
LN	numero	Logaritmo naturale (base "e"). Errore A se l'argomento < =0.
NOT	relazione logica	Se NOT relazione logica è vero la variabile logica è =1, altrimenti è =0.
PEEK	numero	Legge il contenuto del Byte di memoria di indirizzo = a numero arrotondato all'intero più vicino. Errore B se l'argomento non è compreso tra 0 e 65535.
PI	nessuno	Fornisce il numero PI=3.141592653 (p greco)
RND	nessuno	Fornisce il prossimo numero pseudorandom in una sequenza generata usando la formula: (75*(SEED+1) -1)/65536. SEED= al numero contenuto nel contatore dei fotogrammi dello schermo, ad altro se si è usato il comando RAND. Il numero generato è > = 0 e <1.
SGN	numero	Fornisce: 1 se numero negativo 0 se numero = 0 1 se numero positivo.
SIN	numero in radianti	Seno.
SQR	numero	Radice quadrata del numero. Errore B se numero negativo.
TAB	numero	Sposta la posizione di stampa alla colonna indicata dall'argomento. Se il numero è maggiore di 31, la funzione lavora sul resto del numero diviso 32. La linea non viene variata a meno che la colonna richiesta comporti uno spostamento all'indietro. La posizione 0 è la più a sinistra sulla linea.
TAN	numero in radianti	Tangente.
USR	numero	Manda in esecuzione il programma in linguaggio macchina memorizzato a partire dal byte di indirizzo numero (arrotondato all'intero più vicino). Errore B se numero negativo o > 65535.

IL COMANDO PRINT

Il comando PRINT si è arricchito con le due funzioni AT e TAB già descritte; riassumendo si può dire che con la nuova ROM si può dire al calcolatore dove e cosa stampare sul video.

Dopo il comando PRINT si può scrivere una lista comprendente dati da stampare e funzioni di spostamento. Questi elementi devono essere separati dal punto e virgola. Due elementi da stampare possono anche essere separati dalla virgola, ma con la nuova ROM la virgola fa saltare di 16 posizioni (invece che di 8).

Questo non deve essere considerato un impoverimento dato che il comando TAB consente di andare dovunque sulla linea.

Sullo schermo si possono utilizzare solo 22 linee; le ultime due servono per i comandi.

Sono disponibili altri due comandi che riguardano il video; essi sono:
CLS e SCROLL.

Il comando CLS sbianca lo schermo, cioè mette al carattere spazio tutti i byte della memoria di schermo.

Il comando SCROLL sposta di una linea verso l'alto il contenuto dello schermo perdendo la linea superiore e posiziona la stampa all'inizio della linea disponibile in basso.

Esempi:

```
 5   REM PROVA COMANDO SCROLL
10   SCROLL
20   INPUT A$
30   PRINT A$
40   GOTO 10
```

```
 5   REM PROVA COMANDO TAB
10   FOR I=0 to 20
20   PRINT TAB (8*I);I;
30   NEXT I
```

LA GRAFICA

Lo schermo fornisce di norma 22*32 = 704 posizioni di stampa (sono state escluse le ultime due linee). Con i comandi della grafica ognuno di questi 704 punti può essere ulteriormente suddiviso in 4 puntini (PIXEL).

Ogni "puntino" ha due coordinate, x e y, che lo individuano.

Queste coordinate si scrivono abitualmente entro parentesi, così: (5,7); in questo caso si intende riferire un puntino che dista 5 dall'estrema sinistra dello schermo e 7 dal basso. Le coordinate dei puntini negli angoli dello schermo girando in senso antiorario e partendo dal basso sono rispettivamente: (0,0),(63,0),(0,43),(63,43).

I comandi disponibili sono:

PLOT x,Y — scrive un puntino nella posizione x,y
UNPLOT x,y — cancella il puntino nella posizione x,y

Si deve fare attenzione al fatto che le coordinate dei puntini nei comandi PLOT e UNPLOT sono trattate in modo inverso rispetto alla funzione AT.

Nella posizione AT le linee sono numerate da 0 a 21 muovendosi dall'alto verso il basso, e le colonne sono numerate da 0 a 31 muovendosi da sinistra verso destra. Inoltre il primo numero si riferisce alle linee e il secondo alle colonne.

Nei comandi PLOT e UNPLOT le coordinate dei puntini vanno da 0 a 43 muovendosi dal basso verso l'alto e da 0 a 63 muovendosi da sinistra verso destra. Inoltre la prima coordinata si riferisce alle colonne e la seconda alle linee.

Esempi:

```
10   REM GRAFICO FUNZIONE SENO
15   REM TRA 0 E 2PI
20   FOR N = 0 TO 63
30   PLOT N,22+20*SIN(N/32*PI)
40   NEXT N
```

```
10   REM DISEGNA PUNTI A CASO OGNI
20   REM VOLTA CHE SI PREME NEWLINE
30   PLOT INT (RND*64), INT(RND*44)
40   INPUT A$
50   GOTO 30
```

```
10   REM GRAFICO DI SQR TRA 0 E 4
20   FOR N = 0 TO 63
30   PLOT N,20*SQR(N/16)
40   NEXT N
```

Segue un sottogramma che traccia una linea tra due puntini; le coordinate dei due puntini devono essere lette dalla tastiera nel programma principale.

Le coordinate siano (A,B) e (C,D).

```
1000   LET U=C-A
1005   REM U=NUMERO PASSI ORIZZONTALI
1010   LET V=D-B
1015   REM V=NUMERO PASSI VERTICALI
1020   LET D1X=SGN U
1030   LET D1Y=SGN V
1035   REM D1X E D1Y SONO UNO SPOSTAMENTO
1036   REM LUNGO LA DIAGONALE
1040   LET D2X=SGN U
1050   LET D2Y=0
1055   REM D2X E D2Y SONO UNO SPOSTAMENTO
1056   REM VERSO DESTRA O VERSO SINISTRA
1060   LET M=ABS U
1070   LET N=ABS V
1080   IF M>N THEN GOTO 1130
1090   LET D2X=0
1100   LET D2Y=SGN V
1105   REM D2X E D2Y SONO UNO SPOSTAMENTO
1106   REM VERSO L'ALTO O VERSO IL BASSO
1110   LET M=ABS V
1120   LET N=ABS U
1130   REM M È IL MAGGIORE TRA ABSU E ABSV
1140   LET S=INT(M/2)
1145   REM VOGLIAMO MUOVERCI DA (A,B) A
       (C,D) IN M PASSI
1146   REM USANDO: N VOLTE L'INCREMENTO D2
       PER SPOSTAMENTI
1147   REM ORIZZONTALI E VERTICALI E M-N
       VOLTE L'INCREMENTO
1148   REM D1 PER SPOSTAMENTI DIAGONALI,
       DISTRIBUITI IL PIU'
1149   REM UNIFORMEMENTE POSSIBILE
1150   FOR I=1 TO M
1160   PLOT A,B
1170   LET S=S+N
1180   IF S>M THEN GOTO 1230
1190   LET S=S-M
1200   LET A=A+D1X
1210   LET B=B+D1Y
1215   REM SPOSTAMENTO DIAGONALE
1220   GOTO 1250
1230   LET A=A+D2X
1240   LET B=B+D2Y
1245   REM SPOSTAMENTO ORIZZONTALE O
       VERTICALE
1250   NEXT I
1260   RETURN
```

IL CONTROLLO DEL TEMPO

È possibile programmare delle attese calcolate servendosi del comando PAUSE. Si scrive:

PAUSE n

e il programma si ferma per un intervallo di tempo pari al tempo necessario per far apparire n fotogrammi sul video. La velocità dei fotogrammi è di 50 al secondo; con n=32767 si ottiene una pausa di circa 11 minuti. Se n è maggiore di 32767 la pausa corrisponde allo STOP. Si può interrompere la pausa premendo un qualunque tasto.

Al comando PAUSE si deve far seguire una POKE particolare; si deve quindi scrivere:

PAUSE n
POKE 16437,255

questa POKE serve a riposizionare il byte alto del contatore dei fotogrammi.

Con il programma che segue si ottiene un orologio funzionante sul video.

```
5    REM DISEGNAMO L'OROLOGIO
10   FOR N=1 TO 12
20   PRINT AT 10-10*COS(N/6*PI),10+ 10 *SIN(N/6*PI);N
30   NEXT N
35   REM FACCIAMO PARTIRE L'OROLOGIO
40   FOR T=0 TO 10000
45   REM T È IL TEMPO IN SECONDI
50   LET A=T/30*PI
60   LET SX=21+18*SIN A
70   LET SY=22+18*COS A
75   PLOT SX, SY
77   PAUSE 42
79   POKE 16437,255
81   UNPLOT SX,SY
90   NEXT T
```

Le attese non calcolate si ottengono usando il comando STOP e poi CONT per proseguire.

Si può usare il comando INKEY$ per ottenere delle attese controllandone la durata esternamente al programma. Infatti il comando INKEY$ legge dalla tastiera un carattere, se non si preme alcun tasto legge la stringa nulla. Premendo un qualunque tasto e controllandolo a programma si generano delle attese. Il programma che segue prosegue solo se si preme un tasto qualunque:

```
10 IF INKEY$ = " " THEN GOTO 10
20 .....
```

infatti se non si preme alcun tasto e quindi viene letta la stringa

nulla la linea 10 ritorna su se stessa.

Il programma che segue si ferma fino a quando si preme un tasto, se esso è A prosegue alla linea 500, se altro prosegue dalla linea 100:

```
10    IF INKEY$ = " " THEN GOTO 10
20    IF INKEY$ = "A" THEN GOTO 500
30    GOTO 100
```

MEMORIZZAZIONE DI PROGRAMMI E DATI SUL NASTRO

Per memorizzare un programma sul nastro si usa il comando SAVE seguito dal nome del programma scritto tra doppi apici; non è accettata la stringa nulla:

SAVE "PROGRAMMA"

Insieme al programma vengono memorizzate anche le variabili con i loro contenuti. Naturalmente le variabili che vengono inizializzate in frasi di INPUT hanno dei contenuti solo se il programma ha girato.

Il comando SAVE può essere usato anche all'interno di un programma, cioè un programma può memorizzare se stesso dopo aver lavorato. Questo uso di SAVE è raccomandabile se si desidera mantenere all'interno di un programma un archivio di dati, infatti se il programma salva se stesso non si rischia di dimenticare di salvarlo. Si deve programmare uno STOP prima del comando SAVE per consentire le operazioni di attacco del registratore. La procedura consigliata è la seguente:

- 1) Inviare un messaggio al video chiedendo di attaccare il registratore.
- 2) Scrivere una istruzione STOP.
- 3) Scrivere una istruzione SAVE nome-programma.

In fase operativa dopo il messaggio e lo STOP, si scrive CONT senz NEWLINE, si avvia il registratore e si preme NEWLINE.

Il comando SAVE non può essere usato all'interno di un sottoprogramma.

Il comando LOAD carica un programma da nastro; si può scrivere in due modi:

- 1) LOAD
- 2) LOAD "NOME PROGRAMMA"

nel modo 1) viene caricato il primo programma disponibile sul nastro, mentre nel modo 2) viene caricato il programma registrato con il nome specificato.

Se il programma caricato in memoria contiene dati si deve fare attenzione al modo nel quale si manda in esecuzione il programma; infatti RUN distruggerebbe i dati. In questo caso si deve mandare in esecuzione il programma scrivendo:

GOTO numero linea prima istruzione.

Il nome di un programma non deve superare 127 caratteri e può essere dato sotto forma di costante stringa, di variabile stringa o di espressione stringa.

Usando gli accorgimenti consigliati si può ovviare al fatto che con il Sinclair non si possono gestire direttamente files di dati. Se si dispone dell'espansione della RAM a 16K e della nuova ROM il nostro calcolatore diventa molto potente e si possono trattare tranquillamente achivi di dati.

I nastri caricati con il vecchio BASIC non sono compatibili con la nuova ROM e viceversa.

DIFFERENZE RISPETTO ALLO STANDARD

In diversi punti di questo manuale sono state segnalate alcune differenze tra il nuovo BASIC e il Basic Standard. In questo paragrafo si completa l'argomento.

Non è disponibile il comando:

ON X GOTO N1, N2, N3,...NK

si può ottenere quasi lo stesso risultato usando alcuni accorgimenti. Invece di scrivere:

ON X GOTO 100, 200, 300, 400

che ha il significato di mandare: alla linea 100 se X=1,
alla linea 200 se X=2,
alla linea 300 se X=3,
alla linea 400 se X=4;

si può scrivere:

GOTO 100*X

e si ottiene lo stesso risultato.

Non sono disponibili i comandi: READ, DATA e RESTORE per gestire blocchi di dati all'interno di un programma. Ricordiamo che la DATA serve per memorizzare blocchi di dati all'interno di un programma, la READ serve per associare questi dati alle variabili in sequenza e la RESTORE serve per poter ricominciare ad usare i dati dall'inizio del blocco.

Si può ottenere il risultato di avere un gruppo di variabili con determinati contenuti in diversi modi:

- 1) Scrivere una serie di LET variabile = dato.
- 2) Scrivere una serie di istruzioni di lettura dati dall'esterno all'inizio del programma, eventualmente con un ciclo FOR se i nomi delle variabili lo consentono, e poi memorizzare il programma su nastro insieme alle variabili come suggerito nel precedente paragrafo.
- 3) Incorporare i dati in delle REM o in delle stringhe lunghe e poi usare delle routine di smistamento dei dati.

LA STAMPANTE

Il nuovo BASIC consente di usare una stampante collegata al calcolatore e fornisce tre istruzioni per comunicare con essa. Queste istruzioni non sono standard.

LPRINT

Questo comando consente di stampare dati con la stampante, corrisponde al comando PRINT per il video. Bisogna fare attenzione, se si vogliono usare AT e TAB, alle dimensioni orizzontali delle linea di stampa.

LLIST

Questo comando consente di mandare alla stampante liste di programmi, corrisponde al comando LIST per il video.

COPY

Questo comando trasferisce sulla stampante il contenuto del video.

Se volete fermare la stampante mentre lavora, potete usare il tasto BREAK.

ERRORI SEGNALATI DAL SISTEMA

Questo paragrafo sostituisce il capitolo 9 del Manuale ZX80.

Il sistema al termine di ogni lavoro e quando incontra alcune istruzioni particolari segnala lo stato in cui si trova mediante un messaggio che appare nell'angolo in basso a sinistra dello schermo. Abitualmente questo messaggio viene chiamato "messaggio di errore", in realtà sarebbe più corretto chiamarlo "messaggio di stato", dato che quello che viene segnalato non sempre è un errore.

Il messaggio si compone di due parti: n/m.

Dove:

n è il numero della linea dove si è fermato il programma
m è il numero distintivo del messaggio in esadecimale cioè un numero da 0 a F.

TABELLA DEI MESSAGGI

Cod.	Significato	Situazione
0	Tutto è andato bene oppure salto ad una linea con numero maggiore di tutte quelle esistenti. Se si usa CONT in modo immediato il pro-	Qualunque

	gramma prosegue dalla linea n	
1	La variabile di controllo non esiste, cioè non è stata citata nel FOR precedente il NEXT, ma esiste come variabile ordinaria.	NEXT
2	Si è usata una variabile che non era stata definita precedentemente. Se la variabile è singola non c'è stata una frase: LET variabile = espressione. Se la variabile è con indici non c'è la frase di dimensionamento DIM. Se la variabile è di controllo, essa non è stata citata nel FOR e non esiste come variabile ordinaria.	Qualunque
3	Indici fuori dal range stabilito. Se oltre ad essere fuori range l'indice è negativo o >65535 si ha errore di codice B.	Variabili con indice
4	Manca spazio in memoria. Il numero della linea nel messaggio può essere incompletato proprio a causa della mancanza di memoria. Si può avere un programma errato che usa troppa memoria nell'area STACK.	LET,INPUT, DIM,PRINT, LIST,PLOT, UNPLOT,FOR, GOSUB calcolo di funzioni complicate.
5	Non si ha più spazio sul video. Se si usa CONT lo schermo si libera e il lavoro può proseguire.	PRINT, LIST
6	Supero di capacità (overflow) durante un calcolo (risultato in valore assoluto > 10**38).	Qualunque calcolo.
7	Incontra un RETURN, ma non c'è stato prima un GOSUB.	RETURN
8	Si è tentato di usare il comando INPUT in modo immediato.	INPUT
9	È stato eseguito un comando STOP. Se si usa CONT il programma non riesegue la linea del comando STOP, ma prosegue.	STOP
A	Argomento non valido nel calcolo di una funzione.	SQR,LN,ASN, ACS.
B	Numero intero fuori dal range. Se il comando richiede un numero intero, esso viene ottenuto arrotondando il numero decimale in questione all'intero più vicino e in questo modo si esce dal range.	RUN,RAND, POKE,DIM, GOTO,LIST, GOSUB,LLIST PAUSE,PLOT, USR,UNPLOT, CHR$,PEEK. Variabili con ind.
C	Si usa una VAL con stringa non numerica.	VAL
D	1) Programma interrotto dal tasto BREAK.	Alla fine di ogni frase o in LOAD,SAVE LPRINT,LLIST COPY.
	2) Il dato di risposta ad un INPUT numerico inizia con STOP. In questo modo si può interrompere un programma durante l'INPUT.	INPUT
E	Non usato	
F	Il nome del programma usato in SAVE è la stringa nulla.	SAVE

ORGANIZZAZIONE DELLA MEMORIA

Si riporta la nuova mappa della memoria. Continua ad essere valido il principio dei registri che contengono un puntatore all'inizio o alla fine della zona identificata. In tale modo l'utilizzo della memoria non è statico, ma varia a seconda dei tipi di programmi utilizzati.

I primi 125 byte della memoria RAM sono utilizzati dal sistema, nell'Appendice B è riportata la descrizione dei contenuti.

MAPPA DELLA MEMORIA

Utilizzo della zona	**Indirizzi fissi o puntatori**
Variabili del sistema	Indirizzo fisso di inizio 16384
Programma	Indirizzo fisso di inizio 16509
Memoria di schermo (Display File)	Puntatore all'inizio D-FILE (16396)
Area Variabili del Programma	Puntatore all'inizio VARS (16400)
Byte che chiude la zona Variabili	Contenuto del puntatore E-LINE meno uno. Il contenuto del byte è 80 esadecimale (128 dec.)
Area per la linea da scrivere + Area di lavoro	Puntatore all'inizio E-LINE (16404)
Area Stack per il calcolatore	Puntatore all'inizio STKBOT (16410)
Area libera	Puntatore all'inizio STKEND (16412)
Area Stack per il microprocessore	Puntatore registro SP
Area Stack per GOSUB	Puntatore all'inizio ERR-SP (16386)
Area per programmi in linguaggio macchina (USR)	Puntatore all'inizio RAMTOP (16388). Indica il primo byte libero dopo il program.BASIC

Al momento dell'accensione del calcolatore RAMTOP contiene l'indirizzo del primo byte non esistente nella memoria. Se si vogliono introdurre delle routine in linguaggio macchina, accessibili con il comando USR, si può modificare con una POKE il contenuto di RAMTOP e caricare le routine a partire dall'indirizzo contenuto in RAMTOP. Il vantaggio di questa procedura è che il comando NEW non tocca le posizioni di memoria oltre il contenuto di RAMTOP, lo svantaggio è che il contenuto di questo ultimo pezzo di memoria non viene salvato sul nastro quando si memorizza il programma in BASIC con il comando SAVE. Inoltre il programma BASIC non interferisce con la zona di memoria che inizia all'indirizzo contenuto in RAMTOP.

La memoria di schermo inizia dopo il programma all'indirizzo contenuto in D-FILE. La memoria di schermo può contenere 24 linee, ciascuna di 32 caratteri + il carattere NEWLINE. A seconda delle dimensioni della RAM del calcolatore il sistema riserva per lo schermo una zona completa, cioè di 24*33 caratteri, o una zona di dimensioni minori. Se, tenendo conto del valore contenuto in RAMTOP, si ha a disposizione poca memoria il sistema assegna alla memoria di schermo le dimensioni minime e cioè 25 caratteri ed essi alla partenza del sistema o per effetto del comando CLS sono 25 caratteri NEWLINE.

E-LINE contiene l'indirizzo di inizio della parte di memoria dove:

- si sta scrivendo: un comando, una linea di programma o un dato di INPUT
- è disponibile una parte di memoria per lavorare.

STKBOT contiene l'indirizzo di inizio dell'area usata per i calcoli, mentre il registro SP punta all'area stack usata dal microprocessore ZX80.

MEMORIZZAZIONE DELLE LINEE DI PROGRAMMA

Le linee del programma BASIC vengono memorizzate così:

Primo byte	Byte più significativo del numero di linea
Secondo byte	Byte meno significativo del numero di linea
Terzo e quarto bytes	Lunghezza in byte dell'istru-

	zione + 1 per il byte con NEWLINE
Byte successivi	Istruzione
Ultimo byte	NEWLINE corrisponde a 01110110 in binario (76 in esadecimale e 118 in decimale).

MEMORIZZAZIONE DELLE VARIABILI

Le variabili del BASIC hanno tutte nomi simbolici che iniziano con una lettera, i codici ASCII delle lettere sono compresi tra 38 e 63 (tra 26 e 3F in esadecimale) e quindi hanno un codice con solo 6 bit significativi, il primo dei quali a sinistra è sempre 1.
Come si può osservare negli schemi che seguono il sistema gioca sui primi bits del primo carattere del nome per distinguere tra loro i diversi dipi di variabili ed inoltre, in alcuni casi, anche sui primi bits dei caratteri successivi.

VARIABILE NUMERICA CON NOME DI UNA SOLA LETTERA

Primo byte	011 + altri 5 bits codice lettera
Secondo byte	Caratteristica del numero (esponente)
4 bytes	Mantissa del numero con segno

Per ogni variabile di questo tipo sono occupati 6 bytes.

VARIABILE NUMERICA CON NOME LUNGO

Primo byte	1 0 1 + altri 5 bits codice prima lettera
Secondo byte	0 0 + secondo carattere nome
•	
•	
•	
Ultimo byte nome	1 0 + ultimo carattere nome
5 bytes	Valore del numero (1 bytes per esponente + 4 bytes per mantissa)

Per ogni variabile di questo tipo sono occupati 5 bytes + 1 byte per ogni carattere del nome.

VARIABILI NUMERICHE CON INDICE

Primo byte	1 0 0 + ultimi 5 bits codice lettera avendo sostituito il primo bit 1 dello stesso codice con 0.
Secondo e terzo	Numero byte occupati = (5 * numero elementi + (2 * numero-dimensioni)+ 1).
Quarto byte	Numero delle dimensioni.
2 bytes per ogni dimensione	Valore della dimensione. Si ha una coppia di bytes per ogni dimensione.
5 bytes per ogni elem.	Valore dell'elemento: esponente e mantissa.

L'ordine degli elementi è quello che si ottiene facendo variare più rapidamente gli indici più a destra e muovendosi verso sinistra. Esempi:
A(2,3) viene disposto in memoria così:
A(1,1), A(1,2), A(1,3), A(2,1), A(2,2), A(2,3)

B(2,3,4) viene disposto in memoria così:
B(1,1,1), B(1,1,2), B(1,1,3), B(1,1,4), B(1,2,1), B(1,2,2),... B(2,3,3), B(2,3,4)

VARIABILI DI CONTROLLO PER I CICLI FOR-NEXT

Queste variabili possono avere il nome formato da una sola lettera.

Primo byte	1 1 1+ ultimi 5 bits cod. lettera
5 bytes	Valore iniziale variabile di controllo
5 bytes	Valore finale variabile di controllo
5 bytes	Valore dello STEP.
2 bytes	Numero di linea della prima linea dopo il FOR (prima linea delle istruzioni fondamentali del ciclo).

VARIABILI STRINGA

Queste variabili possono avere il nome formato da una sola lettera + il carattere $.

Primo byte	0 1 0 + ultimi 5 bits del codice lettera avendo sostituito il primo bit del codice con 0.
Secondo e terzo byte	Numero dei caratteri della stringa, massimo 32767. Tale numero viene limitato solo dalla disponibilità memoria.
Byte successivi	Testo della stringa. La stringa può essere vuota.

VARIABILI STRINGA CON INDICE

Queste variabili possono avere il nome formato da una sola lettera + il carattere $. Il numero delle dimensioni è a piacere, ma ogni elemento deve avere la stessa dimensione.

Primo byte	1 1 0 + ultimi 5 bits del codice lettera avendo sostituito il primo bit del codice con 0.
Secondo e terzo	Numero bytes occupati = (numero elementi * lunghezza elementi) + 1 + (2 * numero-dimensioni) + 2.
Quarzo byte	Numero dimensioni + 1.
2 bytes per ogni dimens.	Valore della dimensione. Si ha una coppia di bytes per ogni dimensione.
2 bytes	Lunghezza in caratteri di ogni elemento.
Numero bytes necessario per ogni elemento	Elementi uno dopo l'altro in ordine di indice facendo variare più rapidamente l'indice più a destra.

IL LINGUAGGIO MACCHINA

Resta valido quanto detto nel capitolo 12 del Manuale dello ZX80 tenendo condo delle seguenti differenze:

•1) Il risultato dei calcoli eseguiti in linguaggio macchina e passati al programma BASIC tramite il comando USR deve trovarsi nei registri bc.

•2) Al ritorno dal programma in linguaggio macchina il registro IY deve avere valore 4000 esadecimale ed il registro I deve avere valore 1E esadecimale.

•3) La routine di gestione del video usa i registri A', F' IX e R; pertanto se si usa il video il programma in linguaggio macchina non deve usare tali registri.

Si consiglia di scrivere le parti di codice in linguaggio macchina incorporandole byte per byte in delle REM o in delle variabili stringa. All'inizio del programma BASIC il codice macchina può essere trasferito in fondo alla memoria servendosi di RAMTOP. In tale modo si è sicuri che il programma BASIC non interferisce con il codice macchina.

APPENDICE "A" CARATTERI DEL SISTEMA

Riportiamo la nuova tabella dei caratteri del sistema. Rimane completamente valida l'appendice D del Manuale ZX80 riguardo al linguaggio macchina.

Per ogni carattere vengono riportati: il codice decimale, il carattere o delle note esplicative, il codice esadecimale. Nella colonna "carattere o note" si rimanda alle note con *n). In questa stessa colonna sono elencate le parole chiave e le funzioni del linguaggio BASIC.

Nella tabella che segue si usano le seguenti abbreviazioni:
crs. sta per cursore;
inv. sta per in campo inverso.

Si ricorda che ogni carattere è memorizzato in 1 byte (8 bits) e che 1 byte può contenere un numero decimale compreso tra 0 e 255 (e quindi un numero esadecimale compreso tra 0 e FF).

Come si vede dalla tabella non tutte le configurazioni di bits corrispondono a caratteri stampabili.

Codice decim.	Carattere o note	Codice esadec.	Codice decim.	Carattere o note	Codice esadec.
0	spazio	00	48	K	30
1	*1)	01	49	L	31
2	*1)	02	50	M	32
3	*1)	03	51	N	33
4	*1)	04	52	O	34
5	*1)	05	53	P	35
6	*1)	06	54	Q	36
7	*1)	07	55	R	37
8	*1)	08	56	S	38
9	*1)	09	57	T	39
10	*1)	0A	58	U	3A
11	"	0B	59	V	3B
12	*2)	0C	60	W	3C
13	$	0D	61	X	3D
14	:	0E	62	Y	3E
15	?	0F	63	Z	3F
16	(	10	64	RND	40
17	)	11	65	INKEY$	41
18	>	12	66	PI	42
19	<	13	67	*3)	43
20	=	14	68	*3)	44
21	+	15	69	*3)	45
22	−	16	70	*3)	46
23	*	17	71	*3)	47
24	/	18	72	*3)	48
25	;	19	73	*3)	49
26	,	1A	74	*3)	4A
27	.	1B	75	*3)	4B
28	0	1C	76	*3)	4C
29	1	1D	77	*3)	4D
30	2	1E	78	*3)	4E
31	3	1F	79	*3)	4F
32	4	20	80	*3)	50
33	5	21	81	*3)	51
34	6	22	82	*3)	52
35	7	23	83	*3)	53
36	8	24	84	*3)	54
37	9	25	85	*3)	55
38	A	26	86	*3)	56
39	B	27	87	*3)	57
40	C	28	88	*3)	58
41	D	29	89	*3)	59
42	E	2A	90	*3)	5A
43	F	2B	91	*3)	5B
44	G	2C	92	*3)	5C
45	H	2D	93	*3)	5D
46	I	2E	94	*3)	5E
47	J	2F	95	*3)	5F
96	*3)	60	167	B inv.	A7
97	*3)	61	168	C inv.	A8
98	*3)	62	169	D inv.	A9
99	*3)	63	170	E inv.	AA
100	*3)	64	171	F inv.	AB
101	*3)	65	172	G inv.	AC
102	*3)	66	173	H inv.	AD
103	*3)	67	174	I inv.	AE
104	*3)	68	175	J inv.	AF
105	*3)	69	176	K inv.	B0
106	*3)	6A	177	L inv.	B1
107	*3)	6B	178	M inv.	B2
108	*3)	6C	179	N inv.	B3
109	*3)	6D	180	O inv.	B4
110	*3)	6E	181	P inv.	B5
111	*3)	6F	182	Q inv.	B6
112	crs. su'	70	183	R inv.	B7
113	crs.giu'	71	184	S inv.	B8
114	crs.sin.	72	185	T inv.	B9
115	crs. dest.	73	186	U inv.	BA
116	GRAPHICS	74	187	V inv.	BB
117	EDIT	75	188	W inv.	BC
118	NEWLINE	76	189	X inv.	BD
119	RUBOUT	77	190	Y inv.	BE
120	stato K/L	78	191	Z inv.	BF
121	FUNCTION	79	192	*4)	C0
122	*3)	7A	193	AT	C1
123	*3)	7B	194	TAB	C2
124	*3)	7C	195	*3)	C3
125	*3)	7D	196	CODE	C4
126	*3)	7E	197	VAL	C5
127	*3)	7F	198	LEN	C6
128	*1)	80	199	SIN	C7
129	*1)	81	200	COS	C8
130	*1)	82	201	TAN	C9
131	*1)	83	202	ASN	CA
132	*1)	84	203	ACS	CB
133	*1)	85	204	ATN	CC
134	*1)	86	205	LN	CD
135	*3)	87	206	EXP	CE
136	*1)	88	207	INT	CF
137	*1)	89	208	SQR	D0
138	*1)	8A	209	SGN	D1
139	" inv.	8B	210	ABS	D2
140	*2)	8C	211	PEEK	D3
141	$ inv.	8D	211	USR	D4
142	: inv.	8E	213	STR$	D5
143	? inv.	8F	214	CHR$	D6
144	(inv.	90	215	NOT	D7
145	) inv.	91	216	**	D8
146	> inv.	92	217	OR	D9
147	< inv.	93	218	AND	DA
148	= inv.	94	219	<=	DB
149	+ inv.	95	220	>=	DC
150	− inv.	96	221	<>	DD
151	* inv.	97	222	THEN	DE
152	/ inv.	98	223	TO	DF
153	; inv.	99	224	STEP	E0
154	, inv.	9A	225	LPRINT	E1
155	. inv.	9B	226	LLIST	E2
156	0 inv.	9C	227	STOP	E3
157	1 inv.	9D	228	*3)	E4
158	2 inv.	9E	229	*3)	E5
159	3 inv.	9F	230	NEW	E6
160	4 inv.	A0	231	SCROLL	E7
161	5 inv.	A1	232	CONT	E8
162	6 inv.	A2	233	DIM	E9
163	7 inv.	A3	234	REM	EA
164	8 inv.	A4	235	FOR	EB
165	9 inv.	A5	236	GOTO	EC
166	A inv.	A6	237	GOSUB	ED

	zione + 1 per il byte con NEWLINE
Byte successivi	Istruzione
Ultimo byte	NEWLINE corrisponde a 01110110 in binario (76 in esadecimale e 118 in decimale).

MEMORIZZAZIONE DELLE VARIABILI

Le variabili del BASIC hanno tutte nomi simbolici che iniziano con una lettera, i codici ASCII delle lettere sono compresi tra 38 e 63 (tra 26 e 3F in esadecimale) e quindi hanno un codice con solo 6 bit significativi, il primo dei quali a sinistra è sempre 1.
Come si può osservare negli schemi che seguono il sistema gioca sui primi bits del primo carattere del nome per distinguere tra loro i diversi dipi di variabili ed inoltre, in alcuni casi, anche sui primi bits dei caratteri successivi.

VARIABILE NUMERICA CON NOME DI UNA SOLA LETTERA

Primo byte	011 + altri 5 bits codice lettera
Secondo byte	Caratteristica del numero (esponente)
4 bytes	Mantissa del numero con segno

Per ogni variabile di questo tipo sono occupati 6 bytes.

VARIABILE NUMERICA CON NOME LUNGO

Primo byte	1 0 1 + altri 5 bits codice prima lettera
Secondo byte	0 0 + secondo carattere nome
•	
•	
•	
Ultimo byte nome	1 0 + ultimo carattere nome
5 bytes	Valore del numero (1 bytes per esponente + 4 bytes per mantissa)

Per ogni variabile di questo tipo sono occupati 5 bytes + 1 byte per ogni carattere del nome.

VARIABILI NUMERICHE CON INDICE

Primo byte	1 0 0 + ultimi 5 bits codice lettera avendo sostituito il primo bit 1 dello stesso codice con 0.
Secondo e terzo	Numero byte occupati = (5 * numero elementi + (2 * numero-dimensioni)+ 1).
Quarto byte	Numero delle dimensioni.
2 bytes per ogni dimensione	Valore della dimensione. Si ha una coppia di bytes per ogni dimensione.
5 bytes per ogni elem.	Valore dell'elemento: esponente e mantissa.

L'ordine degli elementi è quello che si ottiene facendo variare più rapidamente gli indici più a destra e muovendosi verso sinistra. Esempi:
A(2,3) viene disposto in memoria così:
A(1,1), A(1,2), A(1,3), A(2,1), A(2,2), A(2,3)

B(2,3,4) viene disposto in memoria così:
B(1,1,1), B(1,1,2), B(1,1,3), B(1,1,4), B(1,2,1), B(1,2,2),... B(2,3,3), B(2,3,4)

VARIABILI DI CONTROLLO PER I CICLI FOR-NEXT

Queste variabili possono avere il nome formato da una sola lettera.

Primo byte	1 1 1+ ultimi 5 bits cod. lettera
5 bytes	Valore iniziale variabile di controllo
5 bytes	Valore finale variabile di controllo
5 bytes	Valore dello STEP.
2 bytes	Numero di linea della prima linea dopo il FOR (prima linea delle istruzioni fondamentali del ciclo).

VARIABILI STRINGA

Queste variabili possono avere il nome formato da una sola lettera + il carattere $.

Primo byte	0 1 0 + ultimi 5 bits del codice lettera avendo sostituito il primo bit del codice con 0.
Secondo e terzo byte	Numero dei caratteri della stringa, massimo 32767. Tale numero viene limitato solo dalla disponibilità memoria.
Byte successivi	Testo della stringa. La stringa può essere vuota.

VARIABILI STRINGA CON INDICE

Queste variabili possono avere il nome formato da una sola lettera + il carattere $. Il numero delle dimensioni è a piacere, ma ogni elemento deve avere la stessa dimensione.

Primo byte	1 1 0 + ultimi 5 bits del codice lettera avendo sostituito il primo bit del codice con 0.
Secondo e terzo	Numero bytes occupati = (numero elementi * lunghezza elementi) + 1 + (2 * numero-dimensioni) + 2.
Quarzo byte	Numero dimensioni + 1.
2 bytes per ogni dimens.	Valore della dimensione. Si ha una coppia di bytes per ogni dimensione.
2 bytes	Lunghezza in caratteri di ogni elemento.
Numero bytes necessario per ogni elemento	Elementi uno dopo l'altro in ordine di indice facendo variare più rapidamente l'indice più a destra.

IL LINGUAGGIO MACCHINA

Resta valido quanto detto nel capitolo 12 del Manuale dello ZX80 tenendo condo delle seguenti differenze:

•1) Il risultato dei calcoli eseguiti in linguaggio macchina e passati al programma BASIC tramite il comando USR deve trovarsi nei registri bc.

•2) Al ritorno dal programma in linguaggio macchina il registro IY deve avere valore 4000 esadecimale ed il registro I deve avere valore 1E esadecimale.

•3) La routine di gestione del video usa i registri A', F' IX e R; pertanto se si usa il video il programma in linguaggio macchina non deve usare tali registri.

Si consiglia di scrivere le parti di codice in linguaggio macchina incorporandole byte per byte in delle REM o in delle variabili stringa. All'inizio del programma BASIC il codice macchina può essere trasferito in fondo alla memoria servendosi di RAMTOP. In tale modo si è sicuri che il programma BASIC non interferisce con il codice macchina.

APPENDICE "A" CARATTERI DEL SISTEMA

Riportiamo la nuova tabella dei caratteri del sistema. Rimane completamente valida l'appendice D del Manuale ZX80 riguardo al linguaggio macchina.

Per ogni carattere vengono riportati: il codice decimale, il carattere o delle note esplicative, il codice esadecimale. Nella colonna "carattere o note" si rimanda alle note con *n). In questa stessa colonna sono elencate le parole chiave e le funzioni del linguaggio BASIC.

Nella tabella che segue si usano le seguenti abbreviazioni:
- crs. sta per cursore;
- inv. sta per in campo inverso.

Si ricorda che ogni carattere è memorizzato in 1 byte (8 bits) e che 1 byte può contenere un numero decimale compreso tra 0 e 255 (e quindi un numero esadecimale compreso tra 0 e FF).

Come si vede dalla tabella non tutte le configurazioni di bits corrispondono a caratteri stampabili.

Codice decim.	Carattere o note	Codice esadec.	Codice decim.	Carattere o note	Codice esadec.
0	spazio	00	48	K	30
1	*1)	01	49	L	31
2	*1)	02	50	M	32
3	*1)	03	51	N	33
4	*1)	04	52	O	34
5	*1)	05	53	P	35
6	*1)	06	54	Q	36
7	*1)	07	55	R	37
8	*1)	08	56	S	38
9	*1)	09	57	T	39
10	*1)	0A	58	U	3A
11	"	0B	59	V	3B
12	*2)	0C	60	W	3C
13	$	0D	61	X	3D
14	:	0E	62	Y	3E
15	?	0F	63	Z	3F
16	(	10	64	RND	40
17	)	11	65	INKEY$	41
18	>	12	66	PI	42
19	<	13	67	*3)	43
20	=	14	68	*3)	44
21	+	15	69	*3)	45
22	–	16	70	*3)	46
23	*	17	71	*3)	47
24	/	18	72	*3)	48
25	;	19	73	*3)	49
26	,	1A	74	*3)	4A
27	.	1B	75	*3)	4B
28	0	1C	76	*3)	4C
29	1	1D	77	*3)	4D
30	2	1E	78	*3)	4E
31	3	1F	79	*3)	4F
32	4	20	80	*3)	50
33	5	21	81	*3)	51
34	6	22	82	*3)	52
35	7	23	83	*3)	53
36	8	24	84	*3)	54
37	9	25	85	*3)	55
38	A	26	86	*3)	56
39	B	27	87	*3)	57
40	C	28	88	*3)	58
41	D	29	89	*3)	59
42	E	2A	90	*3)	5A
43	F	2B	91	*3)	5B
44	G	2C	92	*3)	5C
45	H	2D	93	*3)	5D
46	I	2E	94	*3)	5E
47	J	2F	95	*3)	5F
96	*3)	60	167	B inv.	A7
97	*3)	61	168	C inv.	A8
98	*3)	62	169	D inv.	A9
99	*3)	63	170	E inv.	AA
100	*3)	64	171	F inv.	AB
101	*3)	65	172	G inv.	AC
102	*3)	66	173	H inv.	AD
103	*3)	67	174	I inv.	AE
104	*3)	68	175	J inv.	AF
105	*3)	69	176	K inv.	B0
106	*3)	6A	177	L inv.	B1
107	*3)	6B	178	M inv.	B2
108	*3)	6C	179	N inv.	B3
109	*3)	6D	180	O inv.	B4
110	*3)	6E	181	P inv.	B5
111	*3)	6F	182	Q inv.	B6
112	crs. su'	70	183	R inv.	B7
113	crs.giu'	71	184	S inv.	B8
114	crs.sin.	72	185	T inv.	B9
115	crs. dest.	73	186	U inv.	BA
116	GRAPHICS	74	187	V inv.	BB
117	EDIT	75	188	W inv.	BC
118	NEWLINE	76	189	X inv.	BD
119	RUBOUT	77	190	Y inv.	BE
120	stato K/L	78	191	Z inv.	BF
121	FUNCTION	79	192	*4)	C0
122	*3)	7A	193	AT	C1
123	*3)	7B	194	TAB	C2
124	*3)	7C	195	*3)	C3
125	*3)	7D	196	CODE	C4
126	*3)	7E	197	VAL	C5
127	*3)	7F	198	LEN	C6
128	*1)	80	199	SIN	C7
129	*1)	81	200	COS	C8
130	*1)	82	201	TAN	C9
131	*1)	83	202	ASN	CA
132	*1)	84	203	ACS	CB
133	*1)	85	204	ATN	CC
134	*1)	86	205	LN	CD
135	*3)	87	206	EXP	CE
136	*1)	88	207	INT	CF
137	*1)	89	208	SQR	D0
138	*1)	8A	209	SGN	D1
139	" inv.	8B	210	ABS	D2
140	*2)	8C	211	PEEK	D3
141	$ inv.	8D	211	USR	D4
142	: inv.	8E	213	STR$	D5
143	? inv.	8F	214	CHR$	D6
144	(inv.	90	215	NOT	D7
145	) inv.	91	216	**	D8
146	> inv.	92	217	OR	D9
147	< inv.	93	218	AND	DA
148	= inv.	94	219	<=	DB
149	+ inv.	95	220	>=	DC
150	– inv.	96	221	<>	DD
151	* inv.	97	222	THEN	DE
152	/ inv.	98	223	TO	DF
153	; inv.	99	224	STEP	E0
154	, inv.	9A	225	LPRINT	E1
155	. inv.	9B	226	LLIST	E2
156	0 inv.	9C	227	STOP	E3
157	1 inv.	9D	228	*3)	E4
158	2 inv.	9E	229	*3)	E5
159	3 inv.	9F	230	NEW	E6
160	4 inv.	A0	231	SCROLL	E7
161	5 inv.	A1	232	CONT	E8
162	6 inv.	A2	233	DIM	E9
163	7 inv.	A3	234	REM	EA
164	8 inv.	A4	235	FOR	EB
165	9 inv.	A5	236	GOTO	EC
166	A inv.	A6	237	GOSUB	ED

238	INPUT	EE	247	RUN	F7
239	LOAD	EF	248	SAVE	F8
240	LIST	F0	249	RAND	F9
241	LET	F1	250	IF	FA
242	PAUSE	F2	251	CLS	FB
243	NEXT	F3	252	UNPLOT	FC
244	POKE	F4	253	CLEAR	FD
245	PRINT	F5	254	RETURN	FE
246	PLOT	F6	255	COPY	FF

NOTE:

*1) Sono disponibili 22 caratteri grafici, cioè gli stessi dello ZX80 con la vecchia ROM, ma salvo che per lo spazio (CHR$(0)) e lo spazio inverso (quadratino nero CHR$(128)) sono cambiati i codici. Facendo riferimento alla pagina 75 del Manuale ZX80 si elencano le variazioni dei codici dei caratteri grafici:

Vecchio codice	Nuovo codice
4	1
5	2
131	3
6	4
2	5
8	6
7	7
9	8
10	9
11	10
132	129
133	130
3	131
134	132
130	133
136	134
135	135
137	136
138	137
139	138

*2) CHR$(12) rappresenta il carattere lira (L maiuscola tagliata) e CHR$(140) rappresenta lo stesso carattere in campo inverso.

*3) Sono configurazioni di caratteri non usate.

*4) Questo è il carattere " " che rappresentata la stringa nulla (diversa dalla stringa contenente un carattere spazio).

Con il programma:

```
10 INPUT X
15 IF X = 0 THEN GOTO 30
20 PRINT CHR$ (X)
25 GOTO 10
30 STOP
```

Si può ottenere sullo schermo il carattere avente il codice decimale X. Per i caratteri contrassegnati da *3) si ottiene il simbolo ? (punto interrogativo).

APPENDICE "B" VARIABILE DEL SISTEMA

La memoria RAM del sistema inizia con l'indirizzo 16384. I primi 125 byte della RAM sono usati dal sistema pe scopi particolari, la zona utente inizia all'indirizzo 16509.

Nella tabella che segue sono descritti le "Variabili del Sistema", alcune di esse occupano 1 byte, altre 2 byte. Se la variabile occupa 2 byte essa è indirizzata dal byte con indirizzo minore (contrariamente a quanto si sarebbe portati a pensare) e questo è il meno significativo. Per mezzo delle istruzioni PEEK e POKE si possono leggere e scrivere (se è consentito) queste variabili.

Si ricorda che, se la variabile occupa 2 bytes, per scrivere un valore V in essa si deve procedere così:

POKE n+1, INT (V/256) si scrive la parte intera di V/256

POKE n,V-256*INT(V/256) si scrive il resto della divisione precedente

Analogamente per ottenere il valore V di una variabile occupante due bytes (di indirizzo n e n+1 si deve procedere così:

PEEK n + 256*PEEK(n+1)

se si è sicuri che la variabile è positiva; altrimenti per ottenere un valore V corretto si deve procedere così;

LET MSB=PEEK(n+1)
IF MSB > 127 THEN LET MSB = MSB −256
LET V = PEEK N + MSB*256

Nella tabella viene dato un nome mnemonico ad ogni variabile del sistema solo per comodità di riferimento, tali nomi ovviamente non possono essere usati nei programmi BASIC. Le variabili del sistema sono accessibili solo tramite i comandi POKE e PEEK.

Nella colonna "Note" della tabella possono comparire delle lettere maiuscole aventi il seguente significato:

X la variabile non può essere modificata;
N la varianbile può essere modificata senza danno;
S la variabile viene conservata quanso di usa il comando SAVE.

Inoltre, nella stessa colonna, compare un numero che indica quanti bytes sono usati per la variabile o la zona di memoria identificata.

Note	Indir.	Nome	Contenuto
1	16384	ERR-NR	Numero del codice di errore - 1. Di norma contiene 255. Con POKE 16384,n si può forzare uno stop. Se 0<=n<=14 si ottiene uno dei messaggi standard. Se 15<=n<−34 o 99<=n < -127 si hanno messaggi non standard. Se 35<=n<=98 si ottiene un collegamento alla memoria di schermo.
X 1	16385	FLAGS	Indicatori usati dal BASIC.
X 2	16386	ERR-SP	Indirizzo del primo argomento nella STACK area, dopo i GOSUB/RETURN
2	16388	RAMTOP	Indirizzo del primo byte sopra l'area BASIC. Se si fa una POKE in RAMTOP essa ha effetto al primo comando NEW o CLS.
N 1	16390	MODE	Stato del cursore: K, L, F o G.
N 2	16391	PPC	Numero della linea di programma in esecuzione.
S 1	16393	VERSN	0 identifica la versione del nuovo BASIC.
S 2	16394	E-PPC	Numero della linea sulla quale sta il puntatore.
SX2	16396	D-FILE	Vedere organizzazione memoria.

S 2	16398	DF-CC	Indirizzo della posizione di stampa nella memoria di schermo.
SX2	16400	VARS	Vedere organizzazione memoria.
SN2	16402	DEST	Indirizzo della variabile in fase di assegnazione.
SX2	16404	E-LINE	Vedere organizzazione memoria.
SX2	16406	CH-ADD	Indirizzo del prossimo carattere da interpretare usato per PEEK, POKE e NEWLINE.
S 2	16408	X-PTR	Indirizzo del carattere che precede lo stato S del cursore.
SX2	16410	STKBOT	Vedere organizzazione memoria.
SX2	16412	STKEND	Vedere organizzazione memoria.
SN1	16414	BERG	Registro B.
SN2	16415	MEM	Indirizzo area usata come memoria per i calcoli. A volte uguale a MEMBOT.
S 1	16417	non usato	
SX1	16418	DF-SZ	Numero delle linee della parte bassa dello schermo compresa una linea bianca
S 2	16419	S-TOP	Numero delle linee di programma della parte alta dello schermo durante la lista automatica.
S2N	16421	LAST-K	Ultimo tasto premuto.
SN1	16423		Stato della tastiera.
SN1	16424	MARGIN	Numero di linee bianche sopra o sotto il disegno (55)
SX2	16425	NXTLIN	Indirizzo della prossima linea di programma da eseguire.
S 2	16427	OLDPPC	Numero di linea da cui deve partire CONT.
SN1	16429	FLAGX	Flags per usi vari.
SN2	16430	STRLEN	Lunghezza della stringa in fase di assegnazione.
SN2	16432	T-ADDR	Indirizzo dell'elemento seguente nella tabella sintattica.
S 2	16434	SEED	Punto di partenza per RND. Viene preparato da RAND.
S 2	16436	FRAMES	Contatore dei fotogrammi dello schermo. Se il bit 15 è 1, i bits da 0 a 14 sono decrementati per ogni fotogramma. Esso può essere usato come Timer. PAUSE lo usa mettendo a 0 il bit 15 e ponendo nei bits da 0 a 14 la lunghezza della pausa. Quando il conto all'indietro è arrivato a 0 la pausa termina. Se si interrompe la pausa con un qualunque tasto il bit 15 viene rimesso a 1.
S 1	16438	COORDS	Coordinata x dell'ultimo punto ottenuto con PLOT.
S 1	16439		Lo stesso per y.
S 1	16440	PR-CC	Byte meno significativo dell'indirizzo della prossima posizione per LPRINT in PRBUFF.
SX1	16441	S-POSN	Numero della colonna per la posizione di PRINT.
SX1	16442		Numero della linea per PRINT.
S 1	16443	CDFLAG	Flags per usi vari. Il bit 7 è a 1 durante i calcoli e le fasi di stampa al video.
S 33	16444	PEBUFF	Buffer di stampa, 32 caratteri + il carattere NEWLINE.
SN30	16447	MEMBOT	Area di memoria per calcoli. Viene usata per memorizzare quei numeri che possono essere posti nella STACK area.
S 2	16507	non usato.	

APPENDICE "C" SCHEDA NUOVO BASIC

VARIABILI.

Numeriche	Nome: primo carattere alfabetico, altri cifre o lettere o spazi, quanti si vuole. Numeri interi e decimali con precisione tra 9 e 10 cifre e grandezza compresa tra 10 elevato a -39 e 10 elevato a +38.
Stringhe	Nome formato da una lettera seguita da $. Non esiste limite al numero dei caratteri.

COSTANTI

Numeriche	Stesse possibilità che per i contenuti delle variabili numeriche.
Stringhe	Delimitate da apici, lunghezza a piacere possono contenere qualunque carattere salvo gli apici. La stringa nulla è " ". Per ottenere gli apici stampabili all'interno di una stringa si deve usare il carattere "doppio apice" o CHR$ (192).

VARIABILI CON INDICE.

Numeriche	Nome formato da una sola lettera, indici multipli, contenuti come per le variabili numeriche semplici. Si può usare lo stesso nome già usato per una variabile semplice.
Stringhe	Nome formato da una lettera seguita da $, indici multipli, contenuti come per le stringhe semplici, tutti gli elementi devono avere lo stesso numero di caratteri. Il nome usato per una stringa con indici non può essere usato per una stringa senza indici.

Gli indici possono essere costanti, variabili numeriche o espressioni numeriche e vengono arrotondati all'intero più prossimo.

VARIABILI DI CONTROLLO

Numeriche — Nome formato da una sola lettera. Sono usate per controllare i cicli FOR/NEXT e all'interno della variabile viene memorizzato il numero della linea della prima istruzione del ciclo.

ESPRESSIONI.

Operatori aritmetici:

** elevato a. Esempio X**Y, si ha errore B se X negativo. Priorità 10.
— unitario, segno del numero. Priorità 9.
*,/ moltiplicato, diviso. Priorità 8.
+,- addizione, sottrazione. Priorità 6.

Operatori relazionali:

= uguale. Priorità 5.
> maggiore. Priorità 5.
< minore. Priorità 5.
<= min.o ug..Priorità 5.
>= magg. o ug..Priorità 5
<> diverso.Priorità 5

Operatori logici:

NOT negazione. Priorità 4.
AND prodotto logico. Priorità 3.
OR somma logica. Priorità 2.

Gli operatori relazionali e gli operatori logici producono una variabile logica di valore:
1 se condizione vera;
0 se condizione falsa.
Le espressioni logiche e relazionali possono far parte di espressioni aritmetiche, ad esse viene sostituito il valore della variabile logica. Le espressioni vengono valutate da sinistra e destra tenendo conto delle parentesi e delle priorità.

FRASI BASIC.

Nella descrizione delle frasi si usano le seguenti convenzioni:

a	rappresenta una singola lettera
v	rappresenta una variabile
x,y,z	rappresentano espressioni nu meriche
m,n	rappresentano espressioni numeriche arrotondate all'intero più vicino
e	rappresenta una espressione
f	rappresenta una stringa
s	rappresenta una frase BASIC.

Ricordiamo che:

- Si possono usare dovunque espressioni, salvo che per i numeri di linea del programma.
- Tutte le frasi possono essere usate sia in modo immediato che differito (anche se questo può non avere molto significato in alcuni casi) salvo la INPUT che può solo essere usata in modo differito.

Comandi	Commento
CLEAR	Cancella tutte le variabili liberando lo spazio che occupavano.
CLS	Pulisce lo schermo, cioè pone spazi nella memoria di schermo.
CONT	Se il codice di errore è p/q e q<> 0, CONT fa eseguire un: GOTO q se p<>9 GOTO q+1 se p=9.
COPY	Manda sulla stampante, se collegata, una copia dello schermo. Se la stampante non è collegata non ha alcun effetto.
DIMa(n1,..,nk)	Cancella una variabile con indice di nome "a" e la ridefinisce. Non dà errore di ridimensionamento. Tutti gli elementi vengono inizializzati a 0. Errore 4 se manca spazio. Può esistere una variabile singola di nome "a".
DIMa$(n1,..,nk)	Cancella una variabile stringa con indice avente lo stesso nome e la ridefinisce L'ultimo dato in parentesi non è una dimensione, ma la lunghezza di ogni elemento in caratteri. Tutti gli elementi vengono inizializzati con il carattere spazio. Errore 4 se manca spazio. Non può esistere una variabile stringa singola di nome "a$".
FORa=xTOy	significa: FORa=xTOySTEP1.
FORa=xTOySTEPz	Cancella, se esiste, la variabile singola di nome "a" e crea una variabile di controllo di nome "a". x è il valore iniziale di a. y è il valore finale di a. z è l'incremento da usare ad ogni ciclo. L'indirizzo della prima istruzione del ciclo è quello della linea dopo il FOR se lavora in modo differito, della linea precedente il FOR se lavora in modo immediato. Se x>y e z>=0 oppure se x<y e z<=0 salta alla linea del NEXTa. Errore 4 se manca spazio per la variabile di controllo.
GOSUBn	Pone il numero della linea del GOSUB nella Stack area e poi salta alla linea n. Errore 4 se non trova il relativo RETURN.
GOTOn	Salta alla linea n, se la linea n manca salta alla prima linea con numero > n.
IFxTHENs	Se la condizione x è vera (variabile logica uguale a 1) esegue l'istruzione s, altrimenti prosegue dalla linea seguente.
INPUTv	Si ferma in attesa di dati con il cursore ad L per dati numerici e al L tra apici per stringhe. Se si risponde premendo il tasto STOP e si è in attesa di numeri il programma si ferma con errore D. Se si risponde con il tasto STOP all'attesa di stringa viene registrata la parola STOP. Se si usa in modo immediato si ha errore 8. I dati ricevuti in INPUT non restano sul video.
LETv=e	La parola chiave LET è obbligatoria. Una variabile singola non è definita fino a quando non compare in una LET a sinistra di un = o in una frase INPUT. Se v è una variabile stringa con indice o una porzione di stringa (sliced), cioè una variabile stringa di dimensioni predeterminate, vengono troncati a destra i caratteri eccedenti o aggiunti spazi di riempimento.
LIST	Corrisponde a LIST0.

LISTn — Lista il programma sul video a partire dalla linea n. Errore 4 o 5 se la lista non entra nello schermo.

LLIST — Corrisponde a LLIST0.

LLISTn — Come LIST, ma la lista va alla stampante, se la stampante non è collegata non agisce. Se si usa BREAK dà errore D.

LOADf — Cerca un programma di nome f sul nastro e lo carica in memoria insieme alle sue variabili. Se f è la stringa nulla, carica il primo programma che trova sul nastro.
Se si preme BREAK o se si ha un errore sul nastro si ha:
1) se non è ancora stato letto un programma si ferma con errore D;
2) se è stato letto un pezzo di programma esegue automaticamente un NEW.

LPRINT... — Come il comando PRINT, ma invia i dati alla stampante. Viene inviata una linea quando:
1) si passa da una linea alla seguente
2) un comando non termina con, o;
3) una, o un TAB richiede una nuova linea
4) alla fine del programma rimane qualcosa da stampare.
Il comando AT ha significato solo riguardo al numero di colonna. Se si preme BREAK dà errore D. Non ha effetto se manca la stampante.

NEW — Cancella il programma e le variabili, ma non tocca la parte di memoria dopo l'indirizzo contenuto in RAMTOP.

NEXTa — 1) Cerca la variabile di controllo a
2) Aggiunge alla variabile lo STEP
3) Se STEP>=0 e a> limite o se STEP<=0 e a < limite salta alla prima linea del ciclo.
Errore 1 se a non è variabile di controllo. Errore 2 se la variabile a non esiste del tutto.

PAUSEn — Sospende il lavoro per una durata pari all'emissione di n fotogrammi (50 fotogrammi al secondo) o fino a quando viene premuto un qualunque tasto. Se non è 0⩽=n<=65535 si ha errore B. Se n>=32767 si può interrompere la pausa solo premendo un tasto.

PLOTm,n — Scrive il puntino di coordinate m,n e sposta la posizione di stampa dopo il puntino.
0<=m<=63 e 0<=n<=43, altrimenti errore B.

POKEm,n — Scrive il valore n nel byte m. Deve essere: 0<=m<=65535 e -255<=n<=255, altrimenti si ha errore B.

PRINT.... — I "...." stanno per la lista di elementi da stampare. Gli elementi possono essere separati da "," o da ";". Il ";" non modifica la posizione di stampa, mentre la "," sposta la posizione di stampa di 16 posizioni almeno, cioè fa posizionare o in colonna 0 o in colonna 16. Se la lista di stampa non termina con "," o ";" la posizione di stampa si sposta all'inizio della linea seguente.
Gli elementi da stampare possono essere:
1) stringa nulla e quindi niente;
2) una espressione numerica. Viene stampato il segno meno se il valore è negativo. Se il valore assoluto del numero da stampare è:
<=(10**(-5)) o >=(10**13))
esso viene stampato usando la notazione esponenziale. la mantissa viene stampata con al massimo 8 cifre ed il punto decimale dopo la prima. L'esponente viene dopo E, il segno ed è formato da 1 o 2 cifre.
Se il numero è compreso nell'intervallo esso viene stampato con la consueta notazione decimale e con al massimo 8 cifre significative.
3) una espressione stringa. Le parole chiave del linguaggio vengono espanse, il carattere " quote image" viene stampato come un doppio apice. I caratteri che non hanno corrispondenza in stampa vengono stampati come punti interrogativi.
4) ATm,n. Essa agisce sulla posizione di stampa, la linea viene contata a partire dall'alto, la colonna a partire da sinistra. Deve essere: 0<=m<=21, altrimenti si ha errore 5, ma se m=22 o m=23 errore B; 0<=n<=31, altrimenti errore B.
5) TABn. Si considera n modulo 32. Viene modificata la posizione di stampa sulla stessa linea, a meno che questo non comporti spostamenti all'indietro, nel qual caso si passa sulla prossima linea. Deve essere 0<=n<=255, altrimenti errore B.

Se si hanno solo 3K o meno di memoria si ha errore 4 (OUT OF MEMORY).
Errore 5 significa che lo schermo è pieno. In questi casi CONT consente di procedere dopo aver svuotato lo schermo.

RAND — Corrisponde a RAND0.

RANDn — Inizializza la variabile, chiamata SEED, che il sistema usa per generare i numeri pseudo random con la funzione RND. Se n<>0 viene posta SEED=n; se n=0 viene posta SEED uguale al valore di un'altra variabile del sistema, chiamata FRAMES, ed è il contatore dei fotogrammi dello schermo. Si ha errore B se n non è compreso nell'intervallo 0-65535.

REM... — Serve per i commenti, "..." può contenere qualunque carattere meno NEWLINE.

RETURN — Preleva un numero di linea dall'area stack dei GOSUB e salta a

quella linea. Si ha errore 7 se l'area stack è vuota.

RUN
Corrisponde a RUN0.

RUNn
Esegue un CLEAR automatico e fa saltare alla linea n. Se non si vuole il CLEAR si deve usare GOSUBn.

SAVEf
Memorizza un programma e le sue variabili sul nastro con il nome f. Non si può usare SAVE all'interno di un sottoprogramma. Si ha errore F se f è la stringa nulla.

SCROLL
Fa scorrere lo schermo di una linea verso l'alto perdendo la linea più in alto e liberandone una in basso. La linea liberata contiene come primo carattere NEWLINE.

STOP
Fa fermare il programma con codice di errore 9. CONT fa proseguire dalla linea seguente.

TO
Questa parola chiave fa parte del comando FOR/NEXT è viene usata in questo modo per ottenere le substringhe.
Si scrive f(m TO n) per indicare quella parte di stringa f che compresa tra il carattere di posto m e quello di posto n.
I due numeri m ed n devono essere positivi altrimenti di ha errore 3.
Si espongono con degli esempi i casi possibili:
"BELLO"(TO5) dà "BELLO"
"BELLO"(2TO) dà "ELLO"
"BELLO"(TO) dà "BELLO"
"BELLO"(2TO2) dà "E"
"BELLO"(3TO8) dà errore
"BELLO" (5TO4) dà " " stringa nulla.

UNPLOTm,n
Agisce come PLOT, ma cancella il puntino.

FUNZIONI:

Per le funzioni che richiedono un argomento questo può anche essere una espressione. Se l'argomento è una espressione esso deve essere racchiuso tra parentesi, se è una costante o una variabile non è necessario fare uso delle parentesi. L'operando viene indicato con x e si specifica il tipo.

Funz.	Operando	Risultato
ABS	numero	Valore assoluto.
ACS	numero	Arcocoseno in radianti. Errore A se non è -1<=x<=1.
ASN	numero	Arcoseno in radiandi. Errore A se non è -1<=x<=1
AT	Vedere comando PRINT.	
ATN	numero	Arcotangente in radianti.
CHR$	numero	Il carattere di codice x arrotondato all'intero più vicino. Errore B se non è 0<=x<=255.
CODE	stringa	Il codice del primo carattere di x o 0 se x è la stringa nulla.
COS	numero	Coseno. L'operando deve essere in radianti.
EXP	numero	Il numero "e" elevato a x.
INKEY$		(nessun argomento) Legge dalla tastiera il carattere corrispondente al tasto premuto con il cursore nello stato L, se non si preme alcun tasto dà la stringa nulla.
INT	numero	Parte intera del numero troncato.
LEN	stringa	Lunghezza in caratteri della stringa
LN	numero	Logaritmo naturale (in base "e") di x. Errore A se x<=0.
NOT	vedere operatori logici.	
PEEK	numeri	Il valore del byte di indirizzo x, arrotondato al più vicino intero. Errore B se non è 0<=x<=255.
PI		(nessun argomento) Il valore di "pigreco" 3.14159265.
RND		(nessun argomento) Il prossimo numero della sequenza dei numeri pseudo random generati. Il numero generato è compreso tra 0 e 1.
SGN	numero	Segno del numero: -1,0,1.
SIN	numero	Seno. L'operando deve essere in radianti.
SQR	numero	Radice quadrata. Errore B se x<0.
STR$	numero	La stringa di caratteri corrispondente alle cifre del numero con segno se negativo.
TAB	Vedere il comando PRINT.	
TAN	numero	Tangente. L'operando deve essere in radianti.
USR	numero	Va ad eseguire il programma in codice macchina memorizzato in x (arrotondato all'intero più vicino). Al ritorno il risultato si trova nei registri bc. Errore B se non è 0<=x<=65535.
VAL	stringa	Valuta x come espressione numerica. Errore C se la stringa non è numerica.